CLAIMS

1. A controller fault recovery system for an array storage system for storing data objects including at least one parity group having a number N of data blocks and a parity block computed from the N data blocks comprising:

an array of storage devices;

at least N+1 controllers, each controller operably connected to a unique portion of the array of storage devices; and

a distributed file system having at least one input/output manager (IOM) routine for each controller, each IOM routine including:

means for controlling access to the unique portion of the array of storage devices associated with that controller;

means for maintaining a journal reflecting a state of all requests and commands received and issued for that IOM routine; and

means for reviewing the journal and the state of all requests and commands received and issued for that IOM routine in response to a notification that at least one of the IOM routines has experienced an unscheduled stop and publishing any unfinished request and commands for the at least one failed IOM routine.

2. The system of claim 1 wherein if the notification is an external notification that a single IOM has failed, the distributed file system keeps running and the fault recovery system uses a proxy arrangement to recovery, and wherein if the notification is a notification that all of the IOMs experienced an unscheduled stop because more than one IOM has failed, the distributed file system is restarted and the notification occurs internally as part of the restart procedure for each IOM.

3. The system of claim 2 wherein the proxy arrangement assigns a proxy IOM routine for each IOM routine, the proxy IOM routine being an IOM routine different than the assigned IOM routine that includes:

means for monitoring the assigned IOM routine for a failure and, in the event of a failure of only the assigned IOM routine, issuing a notification to all other IOM routines;

means for receiving from all other IOM routines identifications of any unfinished requests or commands for the assigned IOM routine that has failed; and

means for marking in a meta-data block for the assigned IOM routine a state of any data blocks, parity blocks or meta-data blocks associated with the unfinished requests or commands reflecting actions needed for each such block when the assigned IOM routine recovers.

4. The system of claim 2 wherein in the event of a failure of more than one of the IOM routines the distributed file system performs an unscheduled stop of all IOM routines and, upon recovery of at least N of the IOM routines after the unscheduled stop, each IOM routine reviews the journal and state for that IOM routine and the publication of any unfinished requests or commands for that IOM routine from all of the other IOM routines and reconstructs each data block, parity block or metadata block in response, so as to insure that any updates to a block of data and its block of parity are atomic.

5. A computer-implemented method of storing data objects in an array storage system, the data objects including at least one parity group having a number N of data blocks and a parity block computed from the N data blocks, wherein the data objects are stored in the array storage system under software control of a distributed file system having at least a number N+1 of input/output manager (IOM) routines, each IOM routine controlling access to a unique portion of the array storage system and having a plurality of buffers to temporarily store blocks to be transferred in/out of that portion of the array storage system, the method comprising:

(a) receiving a write request at a first IOM to store a new data block and, in response:

(a1) issuing a read command to read an old data block corresponding to the new data block if the old data block is not already in a first buffer in the first IOM, the old data block having a first location in a meta-data structure for the distributed file system that contains an old disk address for the old data block;

(a2) allocating a new disk address for the new data block;

(a3) transferring the new data block into a second buffer in the first IOM;

(a4) issuing a write command to write the new data block from the second buffer at the new disk address;

(a5) making a journal entry that the write command was issued;

(a6) sending an update parity request to a second IOM associated with a parity block of the parity group that includes the old data block;

(a7) determining changes between the old data block and the new data block;

(a8) sending the changes between the old data block and the new data block to the second IOM in response to a request from the second IOM;

(a9) releasing the first buffer in response to a confirmation from the second IOM that the changes between the old data block and the new data block have been received;

(a10) receiving a response to the write command that the new data block has been written;

(a11) changing the old disk address to the new disk address in the first location in the meta-data structure;

(a12) releasing the second buffer;

(a13) sending a message to the second IOM that the write command was completed; and

(a14) deallocating space reserved for the old data block in the meta-data structure and making a journal entry that the write command was completed in

response to receiving a message from the second IOM that the parity update was completed; and

(b) receiving the update parity request at the second IOM and, in response:

(b1) making a journal entry that the update parity request was received;

(b2) sending a request to the first IOM for the changes between the old data block and the new data block;

(b3) issuing a read command to read an old parity block corresponding to the parity block of the parity group that includes the old data block if the parity block is not already in a first buffer in the second IOM, the old parity block having a second location in the meta-data structure for the distributed file system that contains an old disk address for the old parity block;

(b4) receiving in a second buffer in the second IOM the changes between the old data block and the new data block from the first IOM and sending the confirmation that the changes between the old data block and the new data block have been received;

(b5) allocating a new disk address for a new parity block and a third buffer in the second IOM;

(b6) generating the new parity block in the third buffer based on the changes between the old data block and the new data block and the old parity block;

(b7) releasing the first buffer and the second buffer;

(b8) issuing a write command to write the new parity block in the third buffer to the new disk address for the new parity block;

(b9) receiving a response to the write command that the new parity block has been written;

(b10) changing the old disk address for the old parity block to the new disk address for the new parity block in the second location in the meta-data structure;

(b11) making a journal entry that the update parity request was completed;

(b12) sending a message to the first IOM that the update parity request was completed; and

(b13) deallocating space reserved for the old parity block in the meta-data structure in response to receiving a message from the first IOM that the write command was completed.

6. The method of claim 5 wherein the write request to store the new data block includes a toughness value that is an indication of when a write request complete response should be returned based on a user chosen value of the desired balance between response time and reliability and the method further comprises:

if the toughness value requires the new data block and the new parity block to be confirmed as written to the redundant storage array, returning a write request complete response after step (a13); and

if the toughness value requires atomicity of the new data block and the new parity block but does not require the new data block and the new parity block to be confirmed as written to the redundant storage array, returning a write request complete response after step (a8).

7. The method of claim 6 wherein the method further comprises:

if the toughness value does not requires atomicity of the new data block and the new parity block, returning a write request complete response after step (a4).

8. The method of claim 5 wherein step (a7) is accomplished by comparing the contents of the first buffer and the second buffer in the first IOM and generating a delta block that is stored in a third buffer in the first IOM and wherein step (b4) stores the delta block in the second buffer

in the second IOM and step (b5) compares the old parity block in the first buffer in the parity IOM with the delta block in the second buffer of the second IOM and generates the new parity block that is stored in the third buffer in the second IOM.

9. The method of claim 5 wherein the array storage system is not provided with non-volatile random access memory for the journal and wherein at least steps (a5), (a14), (b1) and (b11) further comprise the step of receiving a response that the journal has been flushed to the array storage system.

10. The method of claim 5 wherein at least steps (a11), (a14), (b10) and (b13) further comprise the step of receiving a response that the metadata has been flushed to the array storage system.

11. The method of claim 5 wherein the array storage system is provided with non-volatile random access memory (NVRAM) and the journal is recorded in the NVRAM.

12. The method of claim 5 wherein the method further comprises:

(c) in the event of an unscheduled stop of the array storage system, recovering the data parity group of a data object by reviewing the journal entries for both the first IOM and the second IOM and reconstructing the data block or the parity block in response if necessary.

13. The method of claim 12 wherein the first IOM is a data IOM for a given parity group and the second IOM is a parity IOM for the parity group and wherein step (c) comprises:

(c1) making no changes to the data parity group or the location in the meta-data structure of the old data disk address and the old parity disk address if:

(c1a) no journal entry exist for either the data IOM or the parity IOM for the data parity group;

(c1b) a journal entry exists that the write command was issued but not completed and there is no journal entry that the parity update request was issued;

(c1c) a journal entry exists that the write command was issued but not completed and a journal entry exists that the parity update request was issued but not completed; or

(c1d) a journal entry exists that the write command was completed and a journal entry exists that the parity update request was completed;

(c2) reconstructing the new parity block from the old data block and the new data block if a journal entry exists for the data IOM that the write command was completed and no journal entry exists for the parity IOM that the update parity request was completed; and

(c3) reconstructing changes to the parity from the old parity block and the new parity block and then reconstructing the new data block from the old data block and the changes to the parity if a journal entry exists for the data IOM that the write command was issued but not completed and a journal entry exists for the parity IOM that the update parity request was completed.

14. The method of claim 13 wherein the journal entry of at least steps (a5), (a14), (b1) and (b11) includes the corresponding new disk address and old disk address for the data or parity, respectively, and wherein step (c) further comprises the step of determining whether the new disk address or the old disk address matches a disk address in the meta-data structure for the data or parity, respectively.

15. A computer-implemented method of storing data objects in an array storage system, the data objects including at least one parity group having a number N of data blocks and a parity block computed from the N data blocks, wherein the data objects are stored in the array storage

system under software control of a distributed file system having at least a number N+1 of input/output manager (IOM) routines, each IOM routine controlling access to a unique portion of the array storage system, the method comprising:

at a first IOM:

receiving a write request to store a new block of data for a data object;

issuing an update parity request to a second IOM associated with a parity block corresponding to the new block of data;

issuing a write command to write the new block of data to the storage system; and

receiving a write command complete from the storage system for the new block of data;

at the second IOM:

receiving the update parity request;

computing a new block of parity for the parity group that includes the new block of data; and

issuing a write command to write the new block of parity; and

receiving a write command complete from the storage system for the new block of parity;

for each of the first and second IOM, maintaining a journal of all requests and commands received and issued; and

in the event of an unscheduled stop of the array storage system, recovering the data parity group of the data object by reviewing the journal entries for both the first and second IOM and reconstructing the data block or the parity block in response if necessary.

15. The method of claim 15 wherein each IOM makes a journal entry in response to:

a write data command from a requestor;

an update parity request from another IOM;

a write data command complete from the array storage system;

a write parity command issued in response to the update parity request from another IOM;

a write parity command complete from the array storage system; and

an update parity request complete from another IOM.

16. The method of claim 15 wherein each IOM maintains its own journal.

17. A computer-implemented method of storing data objects in an array storage system, the data objects including at least one parity group having a number N of data blocks and a parity block computed from the N data blocks, wherein the data objects are stored in the array storage system under software control of a distributed file system having at least a number N+1 of input/output manager (IOM) routines, each IOM routine controlling access to a unique portion of the storage system, the method comprising:

at a first IOM:

receiving a write request to store a new block of data for a data object;

issuing an update parity request to a second IOM associated with a parity block corresponding to the new block of data;

issuing a write command to write the new block of data to the storage system; and

receiving a write command complete from the storage system for the new block of data;

at the second IOM:

receiving the update parity request;

computing a new block of parity for the parity group that includes the new block of data; and

issuing a write command to write the new block of parity; and

receiving a write command complete from the storage system for the new block of parity;

at a third IOM that is designated as a proxy for the first IOM:

monitoring requests to the first IOM; and

in the event that the first IOM does not respond to a request, assuming responsibility for responding to the request;

for each of the first and second IOMs, maintaining a journal of all requests and commands received and issued; and

in the event that the first IOM does not respond to a request, recovering the data parity group of the data object by reviewing the journal entries for both the first and second IOM and reconstructing the data block from the parity block.

18. An array storage system for storing data objects including at least one parity group having a number N of data blocks and a parity block computed from the N data blocks comprising:

an array of storage devices; and

a distributed file system having at least a number N+1 of input/output manager (IOM) routines, each IOM routine controlling access to a unique portion of the array of storage devices and maintaining a journal of all requests and commands received and issued for that IOM wherein

a first IOM responds to a write request to store a new block of data for a data object and issues an update parity request to a second IOM associated with a parity block corresponding to the block of data;

the second IOM computes a new block of parity for the parity group that includes the new block of data; and

in the event of an unscheduled stop of a portion of the array of storage devices controlled by either the first IOM or second IOM, the distributed file system reviews the journal for both the first IOM and second IOM once the first

and second IOM are both restarted and reconstructs the data block or the parity block in response so as to insure that any updates to the new block of data or new block of parity are atomic.

19. The system of claim 18 wherein the distributed file system further includes a third IOM designated as a proxy to the first IOM and wherein, in the event that the first IOM does not respond to a request and another IOM publishes that the first IOM is not available, the third IOM assumes responsibility for responding to the request and marks a meta-data file associated with the first IOM that any write requests for data blocks and parity blocks to the first IOM that were handled by the third IOM need to be reconstructed once the first IOM is restarted.

20. The system of claim 19 wherein the third IOM assumes responsibility for the first by starting a new IOM routine on the controller for the third IOM where the new IOM routine acts as a proxy IOM for the first IOM.

21. The system of claim 18 wherein each of the IOMs other than the first IOM respond to the publication that the first IOM is not available by reviewing the journal and state for that IOM and publishing any unfinished requests or commands between that IOM and the first IOM.